



LSEG

Capital Markets

Ratnagopal, Vishkan

Contents

Creating IAM user -	2
Creating EC2 Instance -	3
Setting up Apache Webserver -	4
Creating S3 Bucket -	6
Writing the script-	7
Additional Task -	14

Capital Markets Application Support/DevOps

Creating IAM user -

You can add multiple users at once with the same access type and permissions. [Learn more](#)

User name*

[Add another user](#)

Select AWS access type

Select how these users will access AWS. Access keys and autogenerated passwords are provided in the last step. [Learn more](#)

Access type* ☐ Programmatic access
Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.

☒ **AWS Management Console access**
Enables a **password** that allows users to sign-in to the AWS Management Console.

Console password*

☐ Autogenerated password

☒ Custom password

☐ Show password

Require password reset ☐ User must create a new password at next sign-in
Users automatically get the **IAMUserChangePassword** policy to allow them to change their own password.

* Required

[Cancel](#) [Next: Permissions](#)

Creating user with AWS management console access and S3 bucket.

Users > anish

Summary [Delete user](#)

User ARN [arn:aws:iam::252802875579:user/anish](#)

Path /

Creation time 2021-02-09 19:42 UTC+0530

Permissions Groups (2) Tags Security credentials Access Advisor

Permissions policies (4 policies applied)

[Add permissions](#) [Add inline policy](#)

Policy name	Policy type
Attached directly	
AmazonS3FullAccess	AWS managed policy
AdministratorAccess	AWS managed policy

[Show 3 more](#)

Permissions boundary (not set)

2 roles were created one for general user with read access and other role for admin access this user is provided with admin access for full control.

Creating EC2 Instance -

Step 1: Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Search for an AMI by entering a search term e.g. "Windows"

Quick Start

- My AMIs
- AWS Marketplace
- Community AMIs
- ☐ Free tier only

Amazon Linux 2 AMI (HVM, SSD Volume Type) - ami-04f77aa5870939148 (64-bit x86) / ami-01bdd32fb6f4e4f51 (64-bit Arm)

Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras. This AMI is the successor of the Amazon Linux AMI that is approaching end of life on December 31, 2020 and has been removed from this wizard.

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Red Hat Enterprise Linux 8 (HVM, SSD Volume Type) - ami-044c46b1952ad5861 (64-bit x86) / ami-0ff84575339cfb9cd (64-bit Arm)

Red Hat Enterprise Linux version 8 (HVM), EBS General Purpose (SSD) Volume Type

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

SUSE Linux Enterprise Server 15 SP2 (HVM, SSD Volume Type) - ami-0e413a9954960d83a (64-bit x86) / ami-0fec5879d3ec972f1 (64-bit Arm)

SUSE Linux Enterprise Server 15 Service Pack 2 (HVM), EBS General Purpose (SSD) Volume Type. Amazon EC2 AMI Tools preinstalled, Apache 2.2, MySQL 5.5, PHP 5.3, and Ruby 1.8.7 available.

Feedback English (US) © 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Creating an ec2 instance to host Apache server. Selected red hat enterprise Linux 8 with default configurations as shown below.

Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. Learn more about instance types and how they can meet your computing needs.

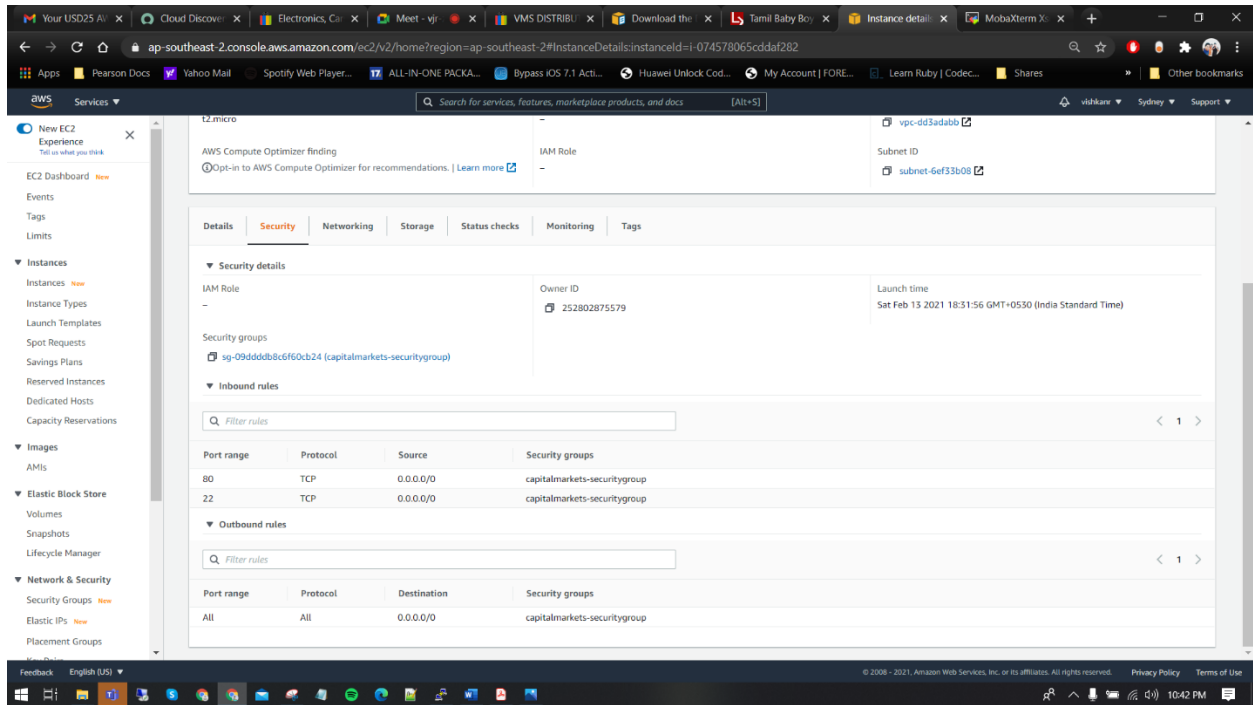
Filter by: All Instance families Current generation Show/Hide Columns

Currently selected: t2.micro (- ECUs, 1 vCPUs, 2.5 GHz, -, 1 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
<input type="checkbox"/>	t2	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
<input checked="" type="checkbox"/>	t2	t2.micro	1	1	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	t2	t2.small	1	2	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	t2	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	t2	t2.large	2	8	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	t2	t2.xlarge	4	16	EBS only	-	Moderate	Yes
<input type="checkbox"/>	t2	t2.2xlarge	8	32	EBS only	-	Moderate	Yes
<input type="checkbox"/>	t3	t3.nano	2	0.5	EBS only	Yes	Up to 5 Gigabit	Yes

Cancel Previous Review and Launch Next: Configure Instance Details

Feedback English (US) © 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use



Security group was created. Inbound and outbound rules were set to publicly accessible to SSH and access the web server as well.

Setting up Apache Webserver -

Following commands are used to set up the web server.

Refresh repository:

```
sudo yum update -y
```

Install php version 7.2, run:

```
sudo yum install php -y
```

Start PHP fpm service:

```
sudo systemctl start php72-php-fpm.service
```

Apache HTTP Server, Install the httpd package:

```
yum install httpd
```

Enable and start the httpd service:

```
systemctl enable --now httpd
```

change the html content:

```
echo "Capital Markets" > /var/www/html/index.html
```

Install Python3:

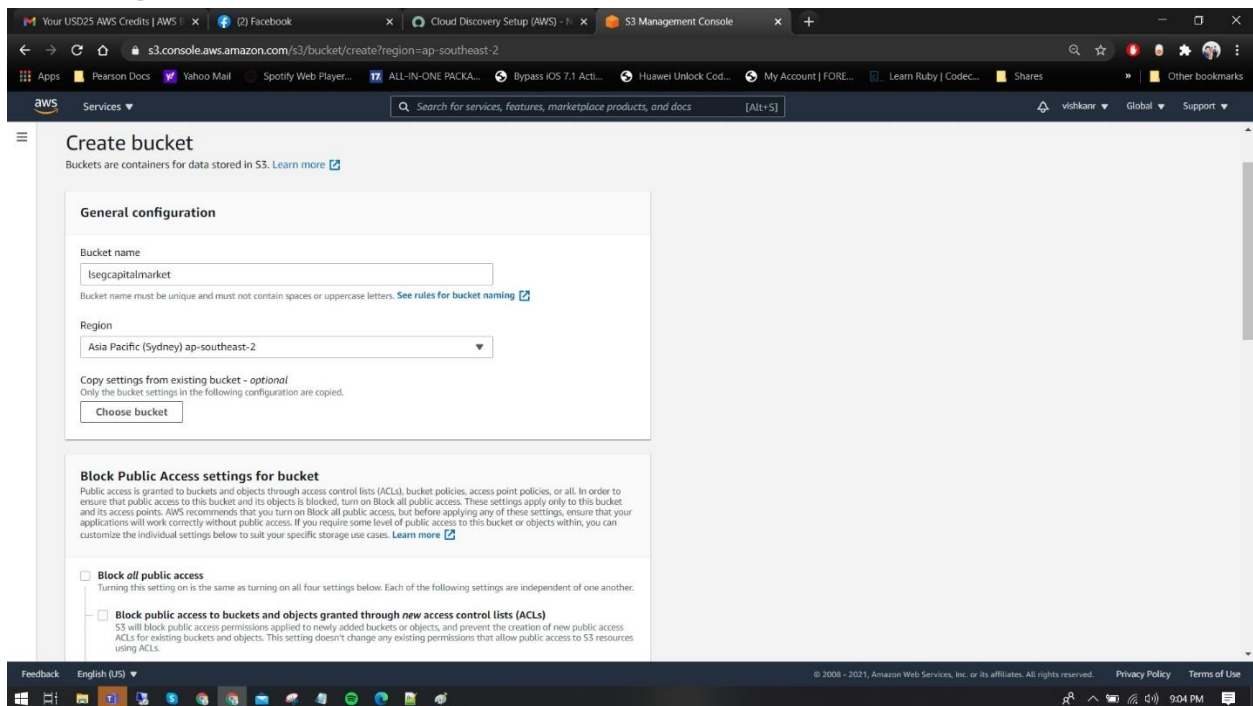
```
sudo yum -y install python3-pip
```

external Import library:

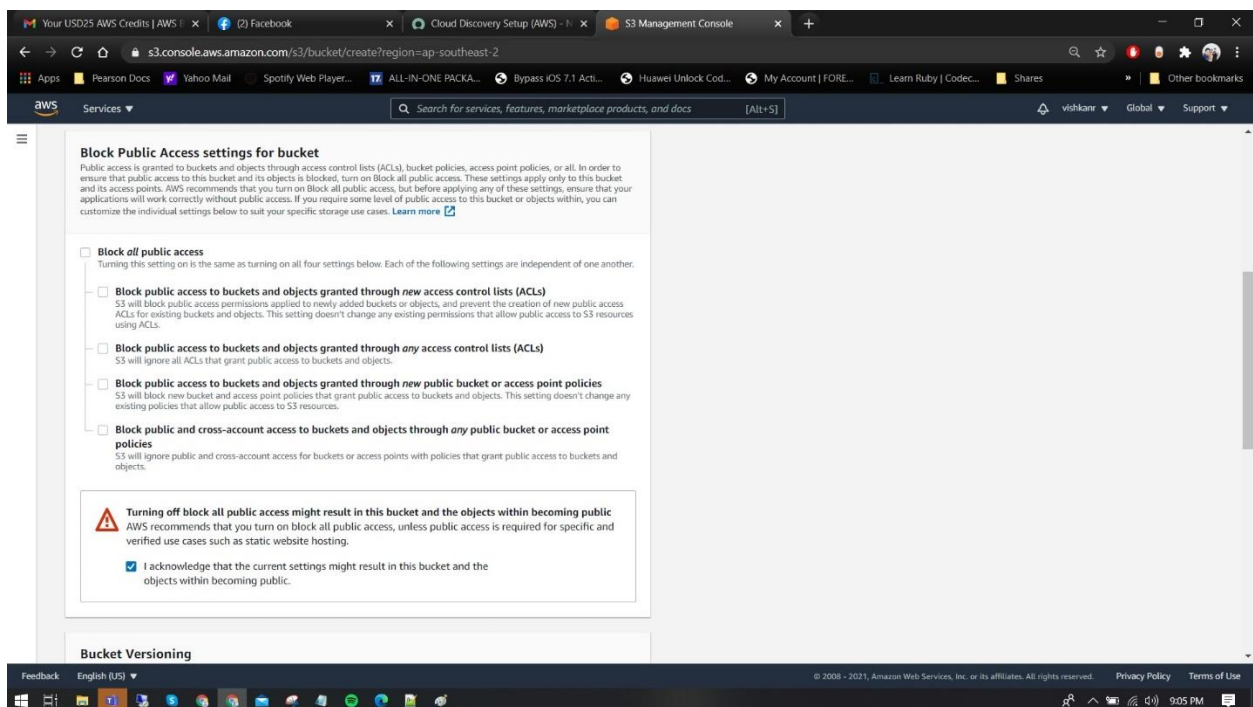
```
import paramiko
```

```
import boto3
```

Creating S3 Bucket -



S3 bucket was created to store log information of the webserver results.



Bucket was given public access to view the log information.

Writing the script-

Following script will test the status of the web server if it's running, script will pass the result and creates a folder in S3 bucket under the following path S3://lsegcapitalmarkets/logs/<date>/ Logs will be created as a text file.

```
#importing library
import paramiko
import datetime

def main():
    # set up environmental variables
    host_name = "13.238.116.70"
    user_name = "ec2-user"
    key_path = "C:/automate/lseg_capital_markets_pem"
    service_name = "httpd"

    # get ssh client object
    ssh_connection = paramiko.SSHClient()

    # remove known host error
    ssh_connection.set_missing_host_key_policy(paramiko.AutoAddPolicy())

    # create connection to host
    ssh_connection.connect(
        hostname=host_name,
        username=user_name,
        key_filename=key_path
    )

    # set up bash commands in string variables
    cmd_service_status = f"systemctl is-active {service_name}"
    cmd_start_httpd = f"sudo systemctl start {service_name}"
    cmd_read_content = f"wget http://{host_name}/ -q -O -"

    # set up date/time variables
    timestamp = datetime.datetime.now().strftime("%Y-%m-%dT%H:%M:%S")
    date = datetime.datetime.now().strftime("%Y-%m-%d")

    # check httpd status
    httpd_status = execute(ssh_connection, cmd_service_status)

    # if httpd is not active then start the service
    if httpd_status != "active":
        execute(ssh_connection, cmd_start_httpd)

    # read web content from the hosted httpd web page
    web_content = execute(ssh_connection, cmd_read_content)

    # write to file and sync to s3
    write_log(ssh_connection, web_content, timestamp, date)

    # close ssh connection
    ssh_connection.close()
```



```

def sync_s3(connection, file_location, date):
    # location of the s3 bucket
    bucket_location = "s3://lsegcapitalmarkets/logs/"

    # command variable for s3 sync
    cmd_copy_bucket = f"aws s3 cp {file_location} {bucket_location}{date}/"
    cmd_verify_bubcket = f"aws s3 ls {bucket_location}{date}/"

    # sync file with s3
    execute(connection, cmd_copy_bucket)

    # verify if files are copied
    execute(connection, cmd_verify_bubcket)

def write_log(connection, content, timestamp, date):
    # text file location to save on local
    file_location = f"logs/{timestamp}.txt"

    # command variables for file write
    cmd_mkdir_log = "mkdir -p logs"
    cmd_write_file = f"echo '{timestamp}: {content}' > {file_location}"

    # create log folder if not exists
    execute(connection, cmd_mkdir_log)

    # write to file
    execute(connection, cmd_write_file)

    # sync file to s3
    sync_s3(connection, file_location, date)

def execute(connection, command) -> str:
    # execute script
    std_in, std_out, std_err = connection.exec_command(command)

    # result of execution from shell standard out
    result = std_out.read().decode().strip()

    # print all outputs
    print(f"{command} - {result}")

    # return result status from script
    return result

if __name__ == '__main__':
    main()

```

Results –

```
Anaconda Prompt (Miniconda3)

(base) C:\Users\vratnvi>cd
C:\Users\vratnvi

(base) C:\Users\vratnvi>cd c:\

(base) c:\>python c:\automate\lseg.py
systemctl is-active httpd - active
wget http://13.238.116.70/ -q -O - - Capital Markets
mkdir -p logs -
echo '2021-02-16T20:59:41: Capital Markets' > logs/2021-02-16T20:59:41.txt -
aws s3 cp logs/2021-02-16T20:59:41.txt s3://lsegcapitalmarkets/logs/2021-02-16/ - Completed 37 Bytes/37 Bytes (697 Bytes
upload: logs/2021-02-16T20:59:41.txt to s3://lsegcapitalmarkets/logs/2021-02-16/2021-02-16T20:59:41.txt
aws s3 ls s3://lsegcapitalmarkets/logs/2021-02-16/ - 2021-02-15 18:39:45      37 2021-02-16T00:09:42.txt
2021-02-15 18:54:38      37 2021-02-16T00:24:35.txt
2021-02-15 21:04:35      37 2021-02-16T02:34:33.txt
2021-02-16 15:23:23      37 2021-02-16T20:53:19.txt
2021-02-16 15:29:45      37 2021-02-16T20:59:41.txt

(base) c:\>
```

By running the first python script it will check the status of the web server (httpd). Then the results echo the date, time and content that was provided in the web server. As an additional effort the log file is created and passed to the S3 bucket for tracking status of the webserver.

Amazon S3 > lsegcapitalmarkets > logs/ > 2021-02-16/

2021-02-16/ Copy S3 URI

Objects Properties

Objects (5)
Objects are the fundamental entities stored in Amazon S3. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

☒ List versions ↻ Delete Actions ▼ Create folder Upload

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	2021-02-16T00:09:42.txt	txt	February 16, 2021, 00:09:45 (UTC+05:30)	37.0 B	Standard
<input type="checkbox"/>	2021-02-16T00:24:35.txt	txt	February 16, 2021, 00:24:38 (UTC+05:30)	37.0 B	Standard
<input type="checkbox"/>	2021-02-16T02:34:33.txt	txt	February 16, 2021, 02:34:35 (UTC+05:30)	37.0 B	Standard
<input type="checkbox"/>	2021-02-16T20:53:19.txt	txt	February 16, 2021, 20:53:23 (UTC+05:30)	37.0 B	Standard
<input type="checkbox"/>	2021-02-16T20:59:41.txt	txt	February 16, 2021, 20:59:45 (UTC+05:30)	37.0 B	Standard

Bellow script will be running externally to the S3 and creates a compressed file according to date of the logs locally and uploads back to S3 as a tar file and send a mail notification if there are no download files for the specified date.

```
import os
import boto3
import datetime
import shutil
import glob
import tarfile
import smtplib
import logging
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText

def main():
    logging.info("starting log archival")

    # current date
    current_date = datetime.datetime.now().strftime("%Y-%m-%d")

    logging.info(f"current date set to {current_date}")

    # s3 variables setup
    resource_name = "s3"
    bucket_name = f"lsegcapitalmarkets"
    archive_url = f"logs_archive/{current_date}"

    # artifact variables setup
    logs_dir = "logs_raw"
    dir_tree = f"{logs_dir}/{current_date}"
    artifact_name = f"{current_date}.tar.gz"

    # mailing variables setup
    username = "vishkanr@gmail.com"
    password = "Civic1999"
    recipient = "vishkanraj@yahoo.com"
    subject = "FAILURE - Logging archive process failed"
    body = f"The log archival process has failed on {current_date} due to "

    logging.info("initialized all required variables")

    # get s3 bucket
    s3 = boto3.resource(resource_name)
    bucket = s3.Bucket(bucket_name)

    logging.info("creating local directories")

    # if local path doesnt exist create it
    if not os.path.exists(dir_tree):
        os.makedirs(dir_tree)
```

```

# keep count of downloaded files
downloaded_count = 0

logging.info("downloading files from s3 bucket")

# iterate all keys
for obj in bucket.objects.all():
    # filename to download taken from the key
    filename = str(obj.key.rsplit('/')[-1]).replace(':', '')

    # download only what is available for specified date
    if current_date in filename:
        bucket.download_file(obj.key, f"{dir_tree}/{filename}")
        downloaded_count += 1

# if download count is greater than 0 upload to s3, else send email to support team
if downloaded_count > 0:
    logging.info("downloading from s3 successful")

    # create tar artifact from all files downloaded
    create_artifact(dir_tree, "txt", artifact_name)

    logging.info("uploading tar archive to s3")

    # upload zip artifact to s3 bucket
    s3.meta.client.upload_file(artifact_name, bucket_name, f"{archive_url}/{artifact_name}")

    logging.info("cleaning directories")

    # remove tar file
    os.remove(f"{artifact_name}")

    # remove raw logs
    shutil.rmtree(logs_dir)
else:
    logging.info("downloading files from s3 failed")

    # update error
    body += "no files being downloaded for set date. Please verify urgently!"

    logging.info("sending email to support team")

    # send email
    generate_email(username, password, recipient, subject, body)

logging.info("completed log archival execution")

```

```

def create_artifact(source_dir, file_extension, artifact_name):
    # open tar creation
    tar = tarfile.open(artifact_name, "w:gz")

    # iterate through source destination looking for specified file extension
    for path in glob.glob(f"{source_dir}/*.{file_extension}"):
        tar.add(path)

    # close tar creation
    tar.close()

def generate_email(username, password, recipient, subject, body):
    # Setup email as multipart MIME
    mail = MIMEMultipart()

    # set email structure
    mail['From'] = username
    mail['To'] = recipient
    mail['Subject'] = subject
    mail.attach(MIMEText(body, 'plain'))

    # setup SMTP session with gmail port and enable security options
    session = smtplib.SMTP('smtp.gmail.com', 587)
    session.starttls()

    # login to email account and send email
    session.login(username, password)
    text = mail.as_string()
    session.sendmail(username, recipient, text)

    # close smtp session
    session.quit()

if __name__ == '__main__':
    # set logging level to info
    logging.basicConfig(level=logging.INFO)

    # start main execution
    main()

```

The script is saved in the following path C:\automate with the “pem” file of the ec2 instance. The region, access key and secret access key are stored in following path for the script to access C:\Users\vratnvi\.aws.

Results –

```
Anaconda Prompt (Miniconda3)
(base) c:\>python C:\automate\lseg_2.py
INFO:root:starting log archival
INFO:root:current date set to 2021-02-17
INFO:root:initialized all required variables
INFO:botocore.credentials:Found credentials in shared credentials file: ~/.aws/credentials
INFO:root:creating local directories
INFO:root:downloading files from s3 bucket
INFO:root:downloading files from s3 failed
INFO:root:sending email to support team
INFO:root:completed log archival execution

(base) c:\>
```

By running the 2nd script, it will create a compressed file within that log file is created with time stamp and content. The log file will be downloaded to local PC (underdevelopment) and a mail is if there are no download files for the specified date.

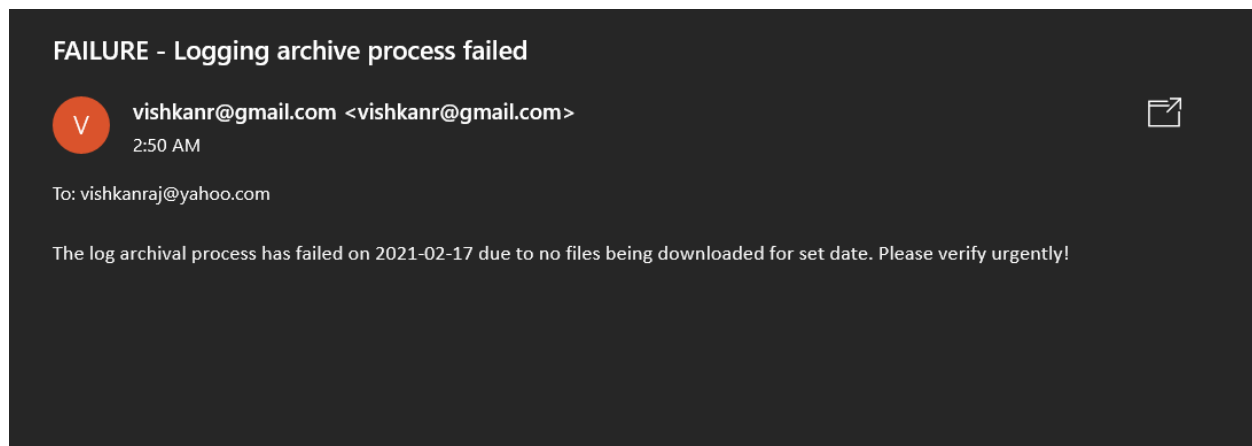
The screenshot displays the Amazon S3 console interface for the bucket 'lsegcapitalmarkets' under the 'logs_archive/' prefix, specifically for the date '2021-02-16/'. The 'Objects' tab is active, showing two objects: '2021-02-16.tar.gz' (297.0 B) and '2021-02-16.tar.gz.tar.gz' (194.0 B). Below the console, a Windows File Explorer window shows the local file structure, including a folder '2021-02-16.tar.gz\logs_raw\2021-02-16 - TAR+GZIP archive, unpacked size 111 bytes' and several text files. A Notepad window is also open, displaying the content of '2021-02-16T00:09:42.txt', which contains the text '2021-02-16T00:09:42: Capital Markets'.

Name	Type	Last modified	Size	Storage class
2021-02-16.tar.gz	gz	February 16, 2021, 02:40:31 (UTC+05:30)	297.0 B	Standard
2021-02-16.tar.gz.tar.gz	gz	February 16, 2021, 02:10:55 (UTC+05:30)	194.0 B	Standard

Name	Size	Packed	Type	Modified	CRC32
2021-02-16T000942.txt	37	?	Text Document	2/16/2021 2:40 ...	
2021-02-16T002435.txt	37	?	Text Document	2/16/2021 2:40 ...	
2021-02-16T023433.txt	37	?	Text Document	2/16/2021 2:40 ...	

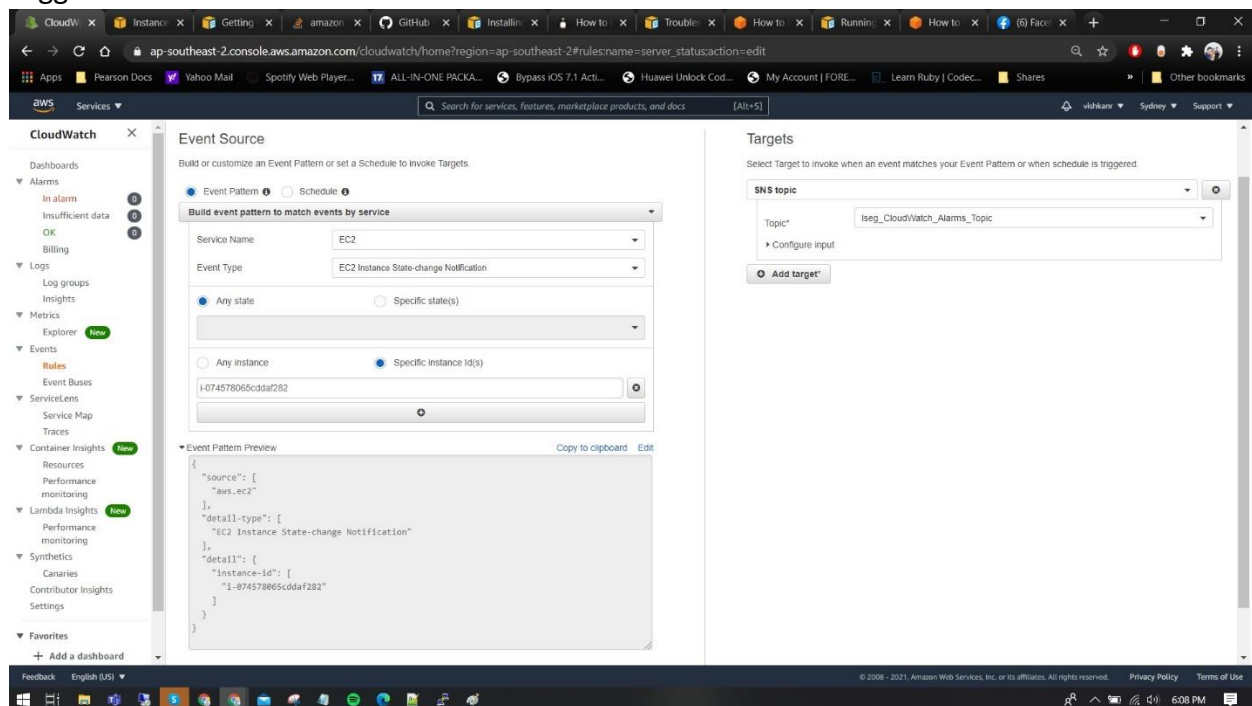
2021-02-16T00:09:42: Capital Markets

Bellow shows the failure if there are no download files for the specified date.



Additional Task -

As an additional task cloud watch was set up if the server goes down or restarts an event is triggered.



SNS was configured to notify the stake holders.

The screenshot displays the Amazon SNS console interface. The left sidebar shows the navigation menu with options like Dashboard, Topics, Subscriptions, and Mobile. The main content area is titled 'Iseg_CloudWatch_Alarms_Topic' and includes a 'Details' section with the following information:

- Name: Iseg_CloudWatch_Alarms_Topic
- ARN: arn:aws:sns:ap-southeast-2:252802875579:Iseg_CloudWatch_Alarms_Topic
- Type: Standard
- Display name: -
- Topic owner: 252802875579

Below the details, there are tabs for 'Subscriptions', 'Access policy', 'Delivery retry policy (HTTP/S)', 'Delivery status logging', 'Encryption', and 'Tags'. The 'Subscriptions' tab is active, showing a list of 3 subscriptions. The list includes a search bar and a table with columns for ID, Endpoint, Status, and Protocol.

ID	Endpoint	Status	Protocol
2e370879-565f-478a-bbb1-b30f66a06d3d	vishkanraj@yahoo.com	Confirmed	EMAIL
81373a25-65b9-4ec7-85de-4dc7f23cedea	+94774900835	Confirmed	SMS
83777911-965e-42ea-620c-ed0c559db50f	+94716320431	Confirmed	SMS

At the bottom of the console, there are buttons for 'Edit', 'Delete', 'Request confirmation', 'Confirm subscription', and 'Create subscription'. The bottom of the screenshot shows the Windows taskbar with various application icons and the system clock indicating 3:34 PM.